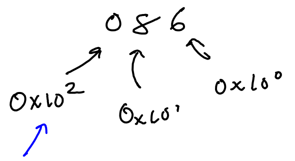
Number Systems & Karnaugh Maps

Wednesday, March 10, 2010

14:59

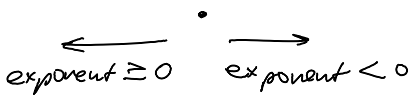
Positional # system:

position of the number defines magnitude:



radix / base ⇒ In this case it's base/radix 10

Radix Point = "Decimal Point"



Binary Addition / Subtraction

Addition is easy

Subtraction is interesting:

carry string

$$\begin{array}{r} 229 \\ -\ \ 46 \\ \hline 183 \end{array} \Longleftrightarrow \begin{array}{rcccccccc} & 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ & 1 & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{0} & \cancel{1} & 0 & 1 \\ - & 0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ \hline & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \end{array}$$

an even # of 1's ⇒ 0

odd # of 0's ⇒ 1

Carry String

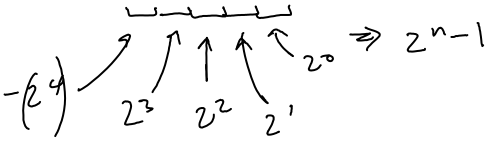
$$\begin{array}{r} 110 \\ 286 \\ +379 \\ \hline 665 \end{array}$$

carry string

⇒ formally declare a carry string, ignore the generated carry values.

⇒ explains why full adder take 3 inputs.

Two's Complement:



35 010011

-35 101100 One's Compliment

$$\begin{array}{r} 35 \\ +-35 \\ \hline 0 \end{array}$$

111111 ← double representation of 0!

-35 110011 ← sign-magnitude

3 more systems excess-m (moveable decimal point)

: for radix=5

0→4 normal

$\bar{4}$→4 signed-digit

BCD - Binary Coded Decimal

⇒ 4 digit binary representation

⇒ only valid representations 0-9

Gray Codes

as we iterate through a sequence of #'s, in a positional # system, we have to do annoying math for overflows/movements.

Gray Codes are <u>NOT</u> a positional # system

Base 10	Grey Code
0	000
1	001
2	011
3	010
4	110
5	111
6	101
7	100

Gray Codes

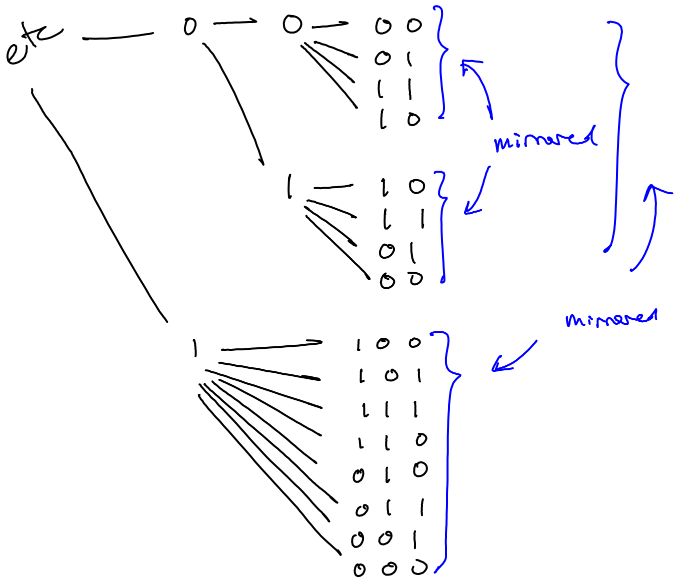
Change only one bit at a time!

2-bit Grey Code

00
01
11
10

} highest order bits look like this

Building a Grey Code



Karnaugh Maps (k-maps)

⇒ 1-bit = boolean variable

⇒ change only one thing at a time

2-variable k-map (looks like a Mendelevian genome map) that represent a particular function with one output

k-map ⇔ f(inputs) = output

each output has a function and a k-map associated with it.

X and Y.

X \ Y	0	1
0	0	0
1	0	1

possible values of Y

possible values of X

Traveling across a row or column holds one of the variables constant.

X or Y

X \ Y	0	1
0	0	1
1	1	1

← if X = 1, we only care about Y

if Y = 1, we only care about X

$$F = XY + X\bar{Y} + \bar{X}Y \Rightarrow X(Y+\bar{Y}) + \bar{X}Y = \boxed{X + \bar{X}Y}$$

OR

$$X\bar{Y} + Y(X+\bar{X}) = \boxed{X\bar{Y} + Y}$$

essentially you're factoring out things.

Thus, we can get

$F = X + Y$

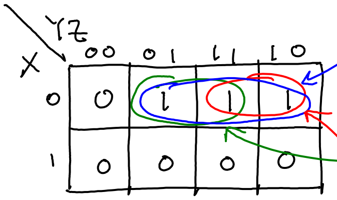
3 Variable k-map

⇒ we have to compress from 3D → 2D, because paper is 2D

X \ YZ	00	01	11	10
0				
1				

note that this is a gray code. we only want to change the one value at a time.

Note that k-maps are torii, you can circle around the boundary

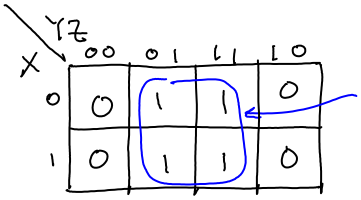


X \ YZ	00	01	11	10
0	0	1	1	1
1	0	0	0	0

invalid circling, because while X is constant, not all values of YZ are covered

valid circling, X is constant, Z is constant, all values of Y covered

valid circling, X is constant, Y is constant all values of Z covered

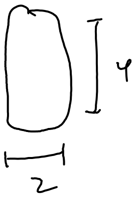


X \ YZ	00	01	11	10
0	0	1	1	0
1	0	1	1	0

valid circling, Z is constant, all values of X, Y are covered

⇒ Takeaway point:

all circling dimensions must be powers of 2 to cover all possible values.



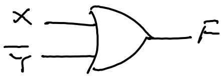
acceptable

Sum of Products

k-map

X \ Y	0	1
0	1	0
1	1	1

$$F = X + \overline{Y}$$

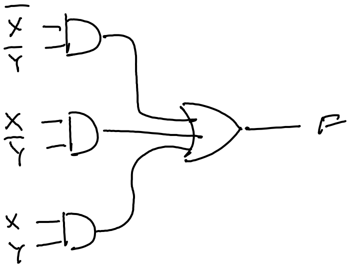


old and busted

X	Y	F
0	0	1
0	1	0
1	0	1
1	1	1

$$F = \overbrace{\overline{X}\,\overline{Y} + X\overline{Y} + XY}^{\text{sum of products}}$$

min term


$\overline{X}$
$\overline{Y}$
X
$\overline{Y}$
X
Y
F